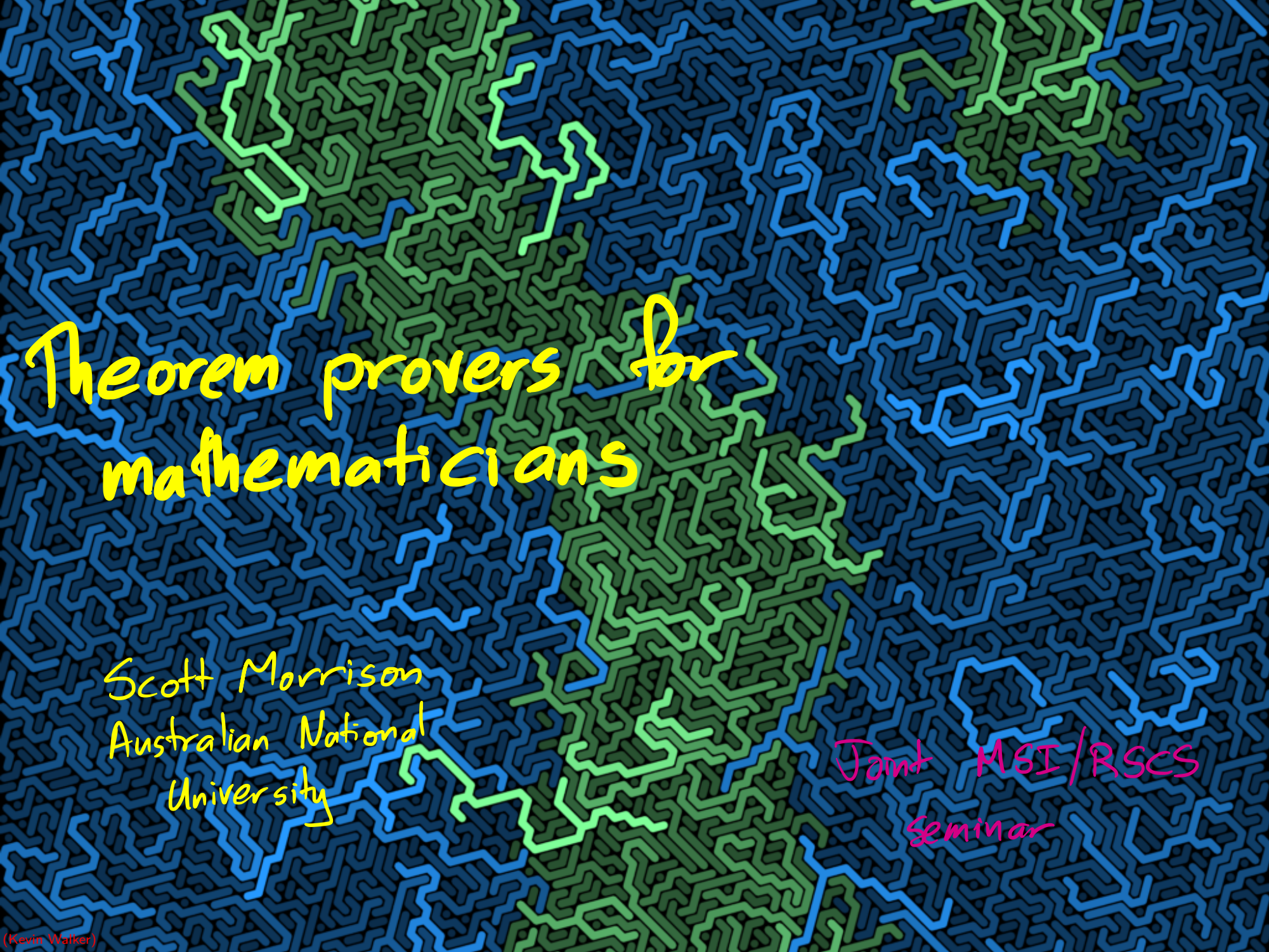

Theorem provers for mathematicians
Scott Morrison
Australian National University
Joint MSI/RSCS seminar
(Kevin Walker)

What is ... an interactive theorem prover?

```
theorem infinitude_of_primes (N : ℕ) : ∃ p ≥ N, prime p :=
begin
  use min_fac (fact N + 1),
  split,
  { by_contradiction h, simp at h,
    back, },
  back
end
```

```
-- We fix a prime number p
parameter (p : Prime)

structure perfectoid_ring (R : Type) [Huber_ring R] extends Tate_ring R : Prop :=
(complete  : is_complete_hausdorff R)
(uniform   : is_uniform R)
(ramified  : ∃ ϖ : pseudo_uniformizer R, ϖ^p | p in R°)
(Frobenius : surjective (Frob R°/p))

/-- Condition for an object of CLVRS to be perfectoid: every point should have an open
neighbourhood isomorphic to Spa(A) for some perfectoid ring A.-/
def is_perfectoid (X : CLVRS) : Prop :=
∀ x : X, ∃ (U : opens X) (A : Huber_pair) [perfectoid_ring A],
  (x ∈ U) ∧ (Spa A ≅ U)

/-- The category of perfectoid spaces.-/
def PerfectoidSpace := {X : CLVRS // is_perfectoid X}
```

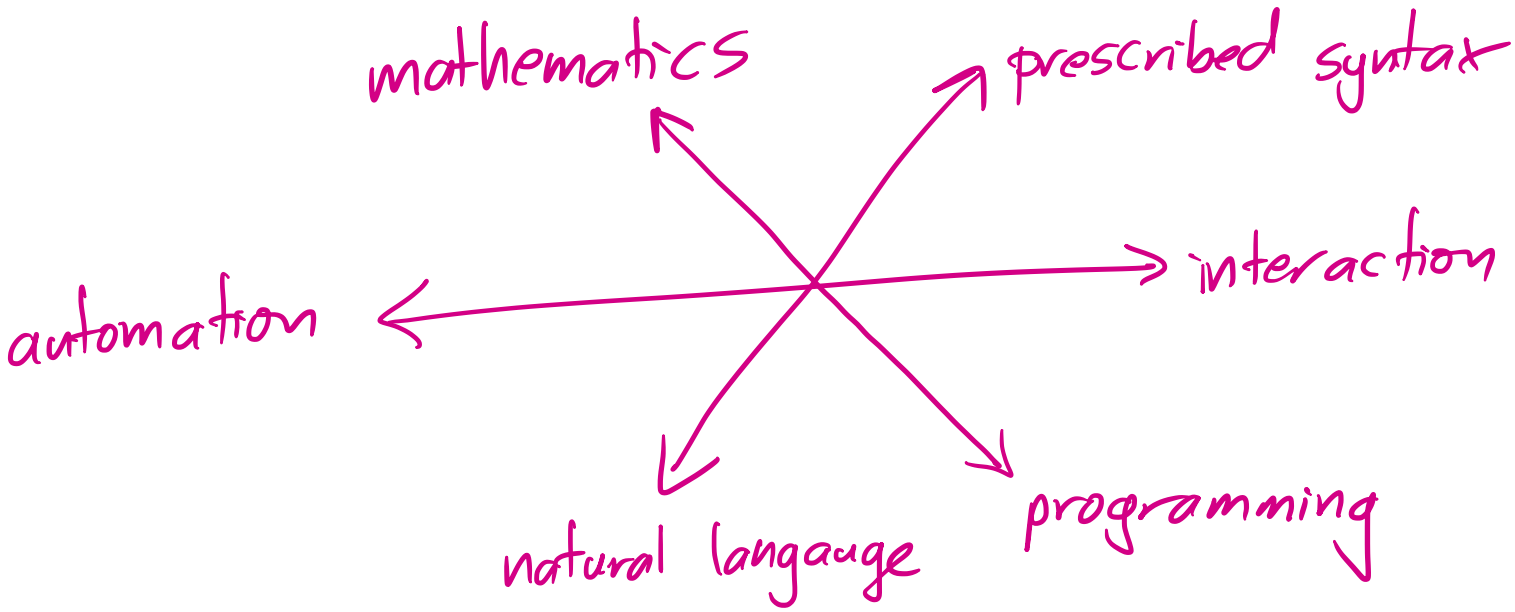
(from Buzzard-Commelin-Massot "Lean perfectoid spaces")

Theorem provers

- use a variety of different logics

  (ZF, higher order logic, dependent type theory, homotopy type theory)

- make different tradeoffs



There's a whole zoo of systems out there!

Isabelle/HOL, Lean, Coq, Agda, Liquid Haskell, Arend

Why?

- a 'proof crisis'?
  (proofs not checked in refereeing?
  instability in the literature due to incomplete publication?
  perhaps even insufficient rigour?)

- deeper understanding?

- better mathematicians?
  perhaps a dream for now, but just as
  computer algebra systems have made us more capable,
  interactive theorem provers with strong automation
  and communicability will help us explore/construct proofs.

Why not?

- Sink effort into particular foundations/proof systems
  - obsolete!?
- It's hard.

  On bad days, the computer is bafflingly obtuse.
  - spell out pedantic details
  - fight with the (dependent) type system
  - slow verification
  - express ideas 'unnaturally' to fit the logic
  - "transport of structure" — a Faustian bargain

- Maybe it's <u>too hard</u>?

Examples

- The four-colour theorem (Gonthier et al) (1976/1997/2005)
  & the Kepler conjecture (Hales et al) (1998/2015)
  - based on verified decision procedures to handle a big case bash!
- The odd order theorem (Gonthier et al) (1962/2012)
  - massive, but 'easy' (character theory...)
- The definition of a perfectoid space (Buzzard-Commelin-Massot) (2012/2019)
  (from Scholze's Fields Medal work)
  - perhaps the conceptually deepest mathematics yet taught to the computer!

```
/-
Perfectoid Spaces

by Kevin Buzzard, Johan Commelin, and Patrick Massot

Definitions in this file follow Scholze's paper: Étale cohomology of diamonds,
specifically Definition 3.1 and 3.19
-/
```

```
structure perfectoid_ring (R : Type) [Huber_ring R] extends Tate_ring R : Prop :=
(complete  : is_complete_hausdorff R)
(uniform   : is_uniform R)
(ramified  : ∃ ϖ : pseudo_uniformizer R, ϖ^p ∣ p in R°)
(Frobenius : surjective (Frob R°/p))

/-
CLVRS ("complete locally valued ringed space") is a category
whose objects are topological spaces with a sheaf of complete topological rings
and an equivalence class of valuation on each stalk, whose support is the unique
maximal ideal of the stalk; in Wedhorn's notes this category is called 𝒱.
A perfectoid space is an object of CLVRS which is locally isomorphic to Spa(A) with
A a perfectoid ring. Note however that CLVRS is a full subcategory of the category
`PreValuedRingedSpace` of topological spaces equipped with a presheaf of topological
rings and a valuation on each stalk, so the isomorphism can be checked in
PreValuedRingedSpace instead, which is what we do.
-/

/-- Condition for an object of CLVRS to be perfectoid: every point should have an open
neighbourhood isomorphic to Spa(A) for some perfectoid ring A.-/
def is_perfectoid (X : CLVRS) : Prop :=
∀ x : X, ∃ (U : opens X) (A : Huber_pair) [perfectoid_ring A],
  (x ∈ U) ∧ (Spa A ≅ U)

/-- The category of perfectoid spaces.-/
def PerfectoidSpace := {X : CLVRS // is_perfectoid X}
```

Interaction & automation

```
def yoneda_long : C ⥤ ((Cᵒᵖ) ⥤ Type v₁) :=
{ obj := λ X,
  { obj := λ Y, (unop Y) → X,
    map := λ Y Y' f g, f.unop ≫ g,
    map_comp' := begin intros, ext1, dsimp, erw [category.assoc] end,
    map_id' := begin intros, ext1, dsimp, erw [category.id_comp] end },
  map := λ X X' f,
    { app := λ Y g, g ≫ f,
      naturality' := begin intros, ext1, dsimp, simp end },
  map_comp' := begin intros, ext1, ext1, dsimp, simp end,
  map_id' := begin intros, ext1, ext1, dsimp, simp end }.

def yoneda_short : C ⥤ ((Cᵒᵖ) ⥤ Type v₁) := ↗ X, ↗ Y, (unop Y) → X.

def yoneda_lemma : (yoneda_pairing C) ≅ (yoneda_evaluation C) :=
{ hom := { app := λ F x, ulift.up ((x.app F.1) (𝟙 (unop F.1))) },
  inv := { app := λ F x, { app := λ X a, (F.2.map a.op) x.down } } }.
```

Outlook for mathematicians

- Theorem provers to help teach proof?
- Students are already using these tools

  (Riesz representation theorem, combinatorial games, CW complexes, braid groups)

- Initial uses in 'real maths' papers.
- Massive libraries to be built (Lean has Noetherian rings, but no holomorphic functions.)
- Lots of automation needed, but lots of low-hanging fruit.
- Collaboration between mathematicians, computers, and computer scientists! (logicians, linguists, ...)